Magic Musicplayer - Development Update #0

Blog - Programming - 03

Luca Hohmann
business@lucahohmann.com

16. April 2019

Abstract

Last week I was undecided on how I will continue with project update blog posts. But after one week of making argument for and against the combination of both projects in one post, I have decided that it is better to keep them separate. So here comes the first development update for the Magic Musicplayer.

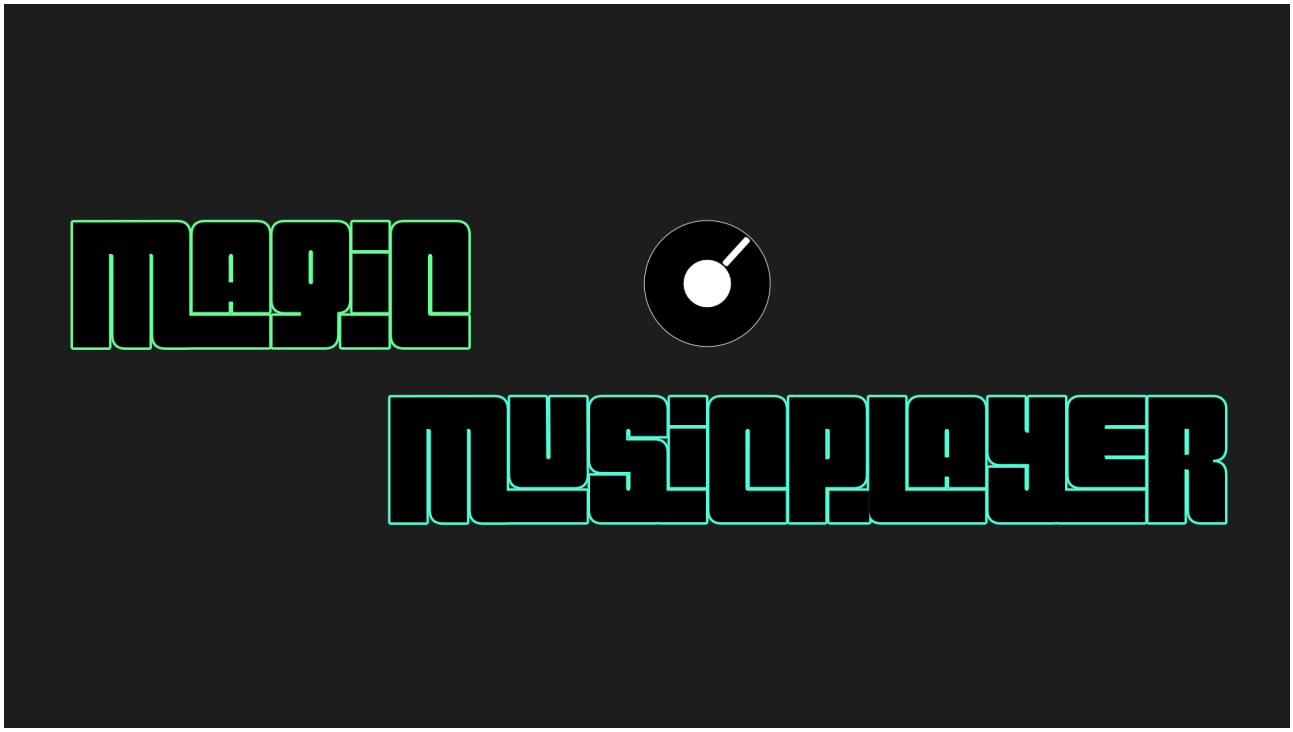

Figure 0.1 MagicMusicplayer logo

1 Introduction

Over the course of the last two week, two very important features found their way into the project. On the one hand the song timeline, which displays the time stamp, the song is currently at, on the other hand, a scrollbar, allowing the user to scroll through the song lists and later album and artist lists, was added.

Because these two features are absolutely necessary, they should work without any flaws. But here comes in a problem, out of my reach. I'll go over this and debate the future for the project, after I have shown and explained the two features. It is also the reason, why only were features were added. Before moving on, I want my current features to work properly and without bugs. But let's get into the features first.

2 The new Features

2.1 Song timeline

The song timeline represents one of the core features, every piece of software, which can replay a mediafile, should provide. It displays a relative time value of the media, in relation to the complete length of the media. So if the song is one minute long, at 0:30 seconds, the timeline marker should be at the center of the timeline. Additionally, the user should be able to move the marker around, in order to change the playback position of the media.

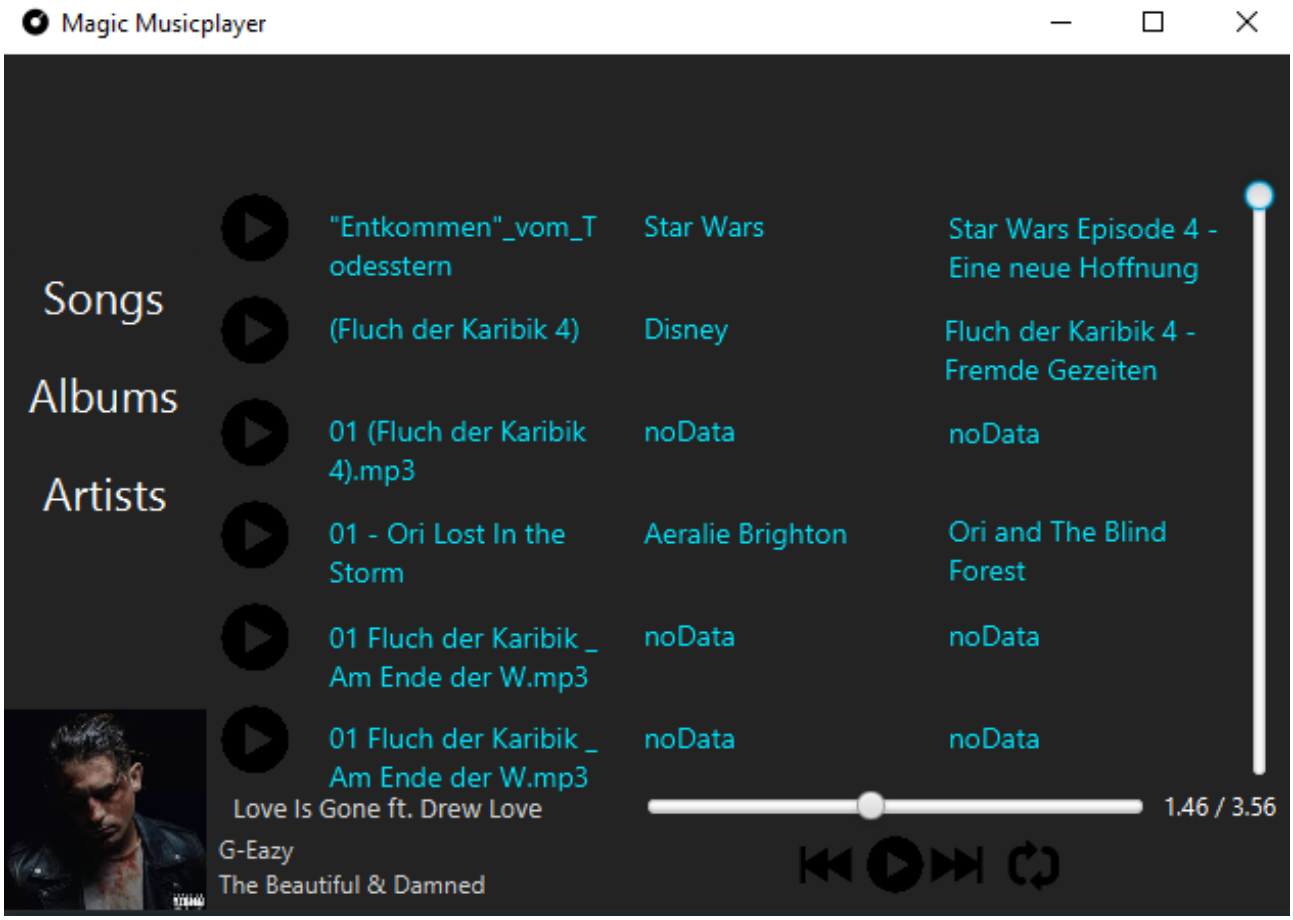

Figure 2.1 Timeline marker is displayed close to the center at minute 1:46, relative to the song length)

In order to implement such a feature, 4 things are needed. First, a fitting UI element, which can represent a timeline with a moving timeline marker. Secondly the possibility to get the current time value of the song as well as the complete length. Thirdly a function, which allows to jump to all time values within the song. The last thing needed, is a new thread, which concurrently checks the marker position and playback time of the song, determines whether the marker was moved and then updates

the marker as well as the playback time accordingly.

As UI element, in my opinion a Slider is the best choice. It is simple, provides the possibility to change all values dynamically and already looks like a timeline. In order to get the current time value of the song, JavaFX provides a nice class construct called "Duration". When you want to access or change values related to the song length or playback position, the Mediaplayer class returns requires you to use the Duration class. The advantage of the class is, that it provides functions which return the time values in seconds, milliseconds, hours and minutes, as well as static functions which return a Duration object with these value formats as parameters. Even tho, convert time values from one format into another, is not very hard, these functions offer very good code readability:

```
mediaPlayer.getCurrentTime().toSeconds();
mediaplayer.getMedia().getDuration().toSeconds()
mediaPlayer.seek(Duration.seconds(20));
```

The first function call returns the current time value of the song in seconds. As the function calls suggest, the second function return the length of the song. The last function should move the playback position of the song to second 20 (important for later: **should**). With these 3 functions everything, that has to be implemented into the external thread, is known. As I said before, the convenience of the Duration class allows to consistently choose a time format, which is suited for the purpose of the application. In this case, I went with seconds, because this format has a unique advantage when it comes to applying all of the values to the UI. So how do we actually use all of these values in the UI?

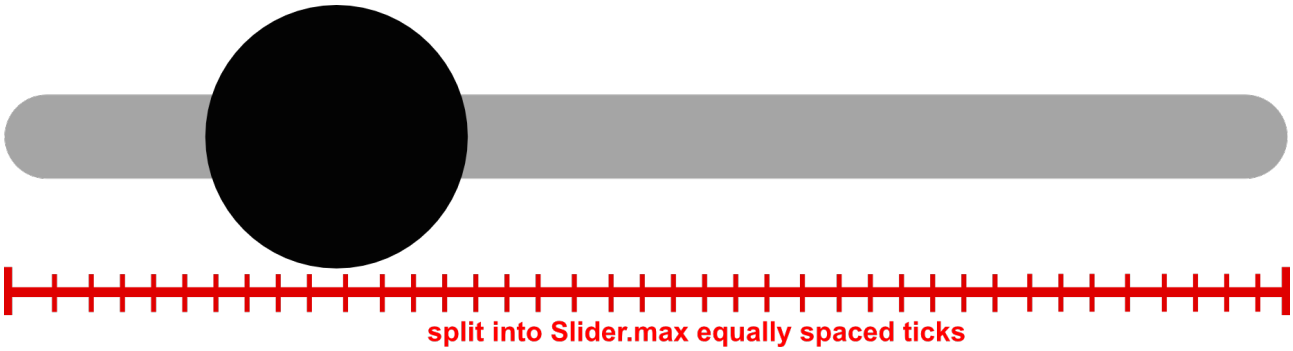

Figure 2.2 Example structure of a slider

As seen in Figure 2.2, the slider's max value determines how many ticks the slider has. It offers the possibility to allow the marker to be on values between the ticks, or only whole number values. For the musicplayer whole number value are enough, because usually milliseconds are not relevant to a user. To put the values from before into context, the song length goes into the slider's max value and the current/seek time into the value variable of the slider, which places the marker.

The remaining part required to connect the mediaplayer with the slider is an external thread. Because JavaFX does not allow a while(true) loop on the main thread, we have to define a special JavaFX thread, called "Service". It acts like a normal thread, but has some special naming conventions, which I do not want to dive into. But lets have a look on what the thread actually does:

```
while mediaplayer.status == playing do
    if slider.gotDragged then
        mediaplayer.seek(Duration.seconds(slider.value));
    else
        slider.value = mediaplayer.getCurrentTime().toSeconds();
    end
    sleep(100); //milliseconds
end
```

In the pseudo code, only 2 unknown things. First being the mediaplayer status, which is pretty self-explanatory. The second new thing is the slider.gotDragged condition. This variable is set to true, when the user drags the marker of the timeline, and therefore changes the playback time of the song. The sleep call is only needed, because otherwise JavaFX still crashes, even tho it is a separate thread.

2.2 Scrollbar

Now that we have seen how the timeline works, the scrollbar (also seen in Figure 2.1), is very easy to explain. It is based on the same idea of a slider, which represents a value, that corresponds to a position in the list. But instead of the playback time, we now change a single index in the UI controller. This value represents the index of the song, which is the highest one, seen in the scrolling UI. According to this value, all song data below is filled. Only preventing out of bounds exceptions, when the slider is at the bottom, had to be implemented additionally. But this is very simple, as the UI is built in a way, that always 6 songs are visible. With this information, we reduce the max value of the slider by 6, and therefore it is not possible to scroll past the last song.

3 Problems with this project

As I have stated at the beginning, there is a major issue with the project, which lies out of my reach. The issue seems to be a bug in JavaFX 8, which is not fully compatible with mp3 files. Mp3 files work as they should, as long as one does not use the seek() function to jump around the audio file. When this function is called, 3 cases can occur. The first one is, that everything works as expected. The second case appears to occur dominantly: When the function is called, the mediaplayer jumps to the time stamp, which was passed via the parameter, but the actual playback time of the song is a few seconds behind. So the function getCurrentTime() returns the correct value, even tho it is completely obvious, that the actual playback is not correct. For example I can jump to the last second before the end of the media file, and the mediaplayer will player multiple seconds before it ends. And the currentTime value will exceed the maximal duration of the audio file, so that you can play 238 of 235 seconds from the song. The third case is even worse, but luckily only appears on a very few files. What happens here is, that while you try to jump withing the first half of the song, the song mediaplayer jumps back to second 0, and start playing from there, but still prints out the original value. And when you jump to a timestamp roughly over 50% of the song, then it jumps to some random point in the first half of the song.

With these two major bugs in place, the usability of such a musicplayer is questionable and the purpose of pushing the project further not very clear. On the other hand, the new features I would implement, are pretty much independent from these bugs.

So overall this is not that an easy situation for me. One thing I will try before deciding on the future of the project is, that I will try to migrate to JavaFX 10 and see whether this might solve the problem.

4 Final words

So if you want to stay updated on this issue, follow me on Twitter: @lucahohmann. You still want to download the current version with both features? Here you can: Github Releases. Until I have decided on how to go on, I will shift my focus onto Treasure Cave and put all effort into this project, so that we can launch it in a reasonable time span.

And yes, thanks for reading, have a good day and see you next time
Luca =)